

Algoritmos y Estructuras de Datos II

Segundo recuperatorio

Miércoles 14 de julio de 2021

Aclaraciones

- El examen es individual en todas sus etapas, y a libro abierto.
- Se responderán consultas por Discord, sólo de interpretación del enunciado, a través del canal ‘Mesa de Docentes’ de la categoría PARCIALES.
- Cualquier aclaración sobre el enunciado se realizará a través del canal ‘anuncios-evaluaciones’.
- El examen durará 5 horas, de 17 a 22hs, y la entrega se realizará a través del Campus hasta esa hora, en forma estricta. Se desestimarán entregas fuera de tiempo, sin excepciones.
- Cada ejercicio se calificará con **Perfecto**, **Aprobado** (que puede ir con ‘–’ o ‘?’), **Regular**, o **Insuficiente**.
- Para aprobar la cursada se deben tener todos los ejercicios aprobados de todos los parciales.
- Sólo se podrá tener un ejercicio con ‘A?’ en la cursada, si no, deberán recuperarse los que hagan falta.

Ej. 1. Elección de estructuras

Sam abrió una nueva tienda de semillas en el barrio. Le ofrecimos diseñar un sistema para hacer el inventario pero se negó con la excusa de que el negocio era muy chico y no lo necesitaba. Un mes después ya tenía cientos de clientes y el trabajo empezó a complicarse. Además, decidió extenderse y empezar a plantar algunas semillas y vender también plantines. Ahora, la necesidad de un sistema que ayude a Sam a manejar el negocio se volvió crítica.

Queremos entonces diseñar un sistema que le permita a Sam conocer en todo momento la cantidad de semillas y de plantines de cada especie que tiene en stock además de llevar un registro de compras y ventas de semillas. Además de las operaciones para comprar semillas y vender tanto semillas como plantines, se requiere contar con la operación `plantar` que reduce la cantidad de semillas y aumenta la cantidad de plantines, teniendo en cuenta que solo la mitad de las semillas que se plantan llegan a germinar.

Se transcribe un fragmento de la especificación:

TAD ESPECIE es NAT

TAD CLIENTE es STRING

TAD PROVEEDOR es STRING

TAD TIENDA

observadores básicos

`inventarioSemillas` : tienda \times especie \rightarrow nat

`inventarioPlantines` : tienda \times especie \rightarrow nat

`ventasSemillas` : tienda \times especie \rightarrow nat

`ventasPlantines` : tienda \times especie \rightarrow nat

generadores

`Iniciar` : \rightarrow tienda

`ComprarSemillas` : tienda \times proveedor \times secu(especie \times nat) \rightarrow tienda

`Vender` : tienda \times cliente \times secu(especie \times nat) \times secu(especie \times nat) \rightarrow tienda

`Plantar` : tienda $t \times$ especie $e \times$ nat $n \rightarrow$ tienda $\{\text{inventarioSemillas}(t, e) \geq n\}$

otras operaciones

`ConvienePlantar` : tienda \rightarrow secu(especie)

RankingVentas : tienda \times nat \longrightarrow secu(especie)
 MáximoStock : tienda \longrightarrow especie
 ...

Fin TAD

Diseñar el módulo TIENDA, de tal modo que provea las siguientes operaciones con las complejidades temporales en peor caso indicadas. Escribir la estructura y detallar los algoritmos de las siguientes operaciones teniendo en cuenta que n es la cantidad de especies distintas que hay a la venta en la tienda.

- **VENDER**(**inout** tienda: tienda, **in** semillas: secuencia(\langle especie, nat \rangle), **in** plantines: secuencia(\langle especie, nat \rangle))
 Sam le vende alguna cantidad 'de semillas y/o plantines a un cliente. Cualquiera de las secuencias puede estar vacía pero no ambas a la vez.
Complejidad: $O(\log(n) * (l_1 + l_2))$ con l_1 y l_2 las longitudes de las secuencias *semillas* y *plantines* respectivamente.
- **PLANTAR**(**inout** tienda: tienda, **in** semillas: especie, **in** cantidad: nat)
 Sam planta *cantidad* semillas de la especie *semillas*.
Complejidad: $O(\log(n))$
- **CONVIENEPLANTAR**(**in** tienda: tienda, **out** semillas: secuencia(especie))
 Secuencia de especies de semillas que conviene plantar. Para que convenga, la cantidad de plantines vendidas de esa especie debe ser al menos el doble que la cantidad de semillas vendidas de esa misma especie.
Complejidad: $O(1)$
- **RANKINGVENTAS**(**in** tienda: tienda, **in** k: nat, **out** semillas: secuencia(especie))
 Secuencia de k especies ordenada de forma decreciente según cuánto se vende. Se sabe que la cantidad k está acotada.
Complejidad: $O(1)$
- **MÁXIMOSTOCK**(**in** tienda: tienda, **out** semillas: especie)
 Devuelve la especie que más hay en stock en la tienda, sumando la cantidad de semillas y de plantines.
Complejidad: $O(1)$

Se pide:

1. Dar una estructura de representación del módulo TIENDA explicando detalladamente qué información se guarda en cada parte, las relaciones entre las partes, y las estructuras de datos subyacentes.
2. Justificar detalladamente de qué manera es posible implementar los algoritmos para cumplir con las complejidades pedidas. Escribir el algoritmo para la operación VENDER.

Para la resolución del ejercicio no está permitido utilizar módulos implementados sobre tablas de *hash* como estructura de representación. Esto responde a dos razones: en primer lugar, el objetivo del ejercicio es que puedan combinar otras estructuras que estudiamos en la materia; en segundo lugar, considerar que los costos de inserción, búsqueda y borrado en un diccionario o conjunto implementado sobre una tabla de *hash* no son constantes en peor caso, y por lo tanto seguramente excedan los costos necesarios para resolver los ejercicios.

Donde se menciona **secuencia** en las definiciones de las operaciones anteriores se debe usar alguno de los módulos vistos que se explican con SECUENCIA, de acuerdo a lo que está en el apunte de Módulos Básicos, o definir uno nuevo que también se explique con ese TAD.

Ej. 2. Ordenamiento

Un coleccionista de figuras de acción de películas quiere ordenar su vitrina. Los objetos los clasifica según nombre del personaje, grupo de películas y número de película. Puede tener varias figuras de acción repetidas. Los nombres de los personajes y de las películas tienen una longitud no mayor a 100, y pueden contener cualquier tipo de caracteres (podemos suponer que conocemos la relación de orden de estos caracteres). Los números de películas son enteros entre 1 y p inclusive. Se sabe que no existen más de 50 personajes por película.

El coleccionista desea ordenar primero por grupo, luego por número de película y finalmente por personaje. Si tiene figuras repetidas, desea conservar solo una de ellas.

Por ejemplo, si $G = [\text{"Star wars"}, \text{"El señor de los anillos"}, \text{"Volver al futuro"}]$, $p = 7$ y

$$C = [\begin{array}{l} (\text{"Leia Organa/Skywalker"}, \text{"Star wars"}, 7) \\ (\text{"Frodo Baggins"}, \text{"El señor de los anillos"}, 2) \\ (\text{"Samwise Gamgee (Sam)"}, \text{"El señor de los anillos"}, 1) \\ (\text{"Luke Skywalker"}, \text{"Star wars"}, 4) \\ (\text{"Emmett «Doc» Brown"}, \text{"Volver al futuro"}, 1) \\ (\text{"Leia Organa/Skywalker"}, \text{"Star wars"}, 4) \\ (\text{"Samwise Gamgee (Sam)"}, \text{"El señor de los anillos"}, 1) \\ (\text{"Emmett «Doc» Brown"}, \text{"Volver al futuro"}, 1) \end{array}]$$

la salida debe ser

$$C_{ord} = [\begin{array}{l} (\text{"Samwise Gamgee (Sam)"}, \text{"El señor de los anillos"}, 1) \\ (\text{"Frodo Baggins"}, \text{"El señor de los anillos"}, 2) \\ (\text{"Leia Organa/Skywalker"}, \text{"Star wars"}, 4) \\ (\text{"Luke Skywalker"}, \text{"Star wars"}, 4) \\ (\text{"Leia Organa/Skywalker"}, \text{"Star wars"}, 7) \\ (\text{"Emmett «Doc» Brown"}, \text{"Volver al futuro"}, 1) \end{array}]$$

Dar un algoritmo de tiempo $O(g \log g + gp + n \log g)$ que ordene la colección y elimine los repetidos, donde n es la cantidad de elementos de C y g es la cantidad de elementos de G . Explicar por qué el algoritmo propuesto cumple con todo lo pedido.

Aclaración: se puede asumir que la comparación entre dos caracteres es constante.

Ej. 3. Dividir y conquistar

Dar un algoritmo que, dado un arreglo ordenado de n números naturales, todos distintos entre sí, encuentre en tiempo $O(\log n)$ el menor número natural que no pertenece al arreglo.

Explicar por qué el algoritmo hace lo que debe hacer y demostrar por qué cumple con la cota de complejidad pedida.